[0070] The coefficients An . . . A**0** used in the polynomial function must be determined quantitatively based on actual sample measurements. One suitable known method used for calculating coefficients based on measurements data is the Levenberg—Marquardt method, which requires Sensor calibration data. This data can be stored in a regular text file where each line keeps values of separate measurement sample, where for example, the first number on the line is the force value in mg (x variable of our function) and the second—the output value of sensor. For example:

Mg:	mV:
80901,	845
81002,	855
81050,	862

[0071] Application of LM is a one time operation which can be implemented in different ways depending on the requirements of the touch screen product and production process. Specifically, the LM procedure reads the calibration data for each sensor, calculates the coefficients for the above polynomial function of each sensor and stores the result to a file.

[0072] The Linearization & Homogenization of Sensor Data function loads these coefficients from the file and uses them in runtime in the polynomial function to calculate the force value for each sensor based on raw sensor output in runtime. The compensated values (in mg) are then used by the position determination software **16** for coordinate determination, which coordinates are passed into the touchscreen application programming interface (API) **18**.

[0073] One skilled in the art should readily understand that the LM coefficient calculation may be integrated into the manufacturing process in various ways. For example, as product is produced a calibration station in the production line may produce a known force at predetermined places on the touch screen. Force and measurements are recorded and the coordinate data file is stored. Alternately, each sensor may be measured during sensor production, or just a few random samples may be measured per production batch and the average coefficients applied to all sensors in the batch.

[0074] The same homogenization module described above is also capable of adjusting for different sensor **10** designs, thus allowing for the same software to be used for different sensor types. The Linearization & Homogenization of Sensor Data function may be implemented as one executable program containing several embedded functions for initialization and compensation itself. Examples of the functions in the runtime sensor compensation program are provided below:

[0075] fsr_compensation_init—this function, **131** is illustrated in the block diagram of FIG. **8**. The function loads up coefficients from coefficient output file, **121**, for each sensor to use them in the polynomial function. It returns status of operation—flagged a success if all coefficients were loaded and a failure otherwise.

[0076] fsr_compensate—this is function **132** in FIG. **8**, which takes sensors output values (in mV) as arguments and calculates force values (in mg) of each sensor. It uses internal function lin_function—the implementation of the polynomial function. The degree of the polynomial function is calculated upon amount of read coefficients: degree=coefs_number−1.

[0077] If the calculation was successful then input arguments are updated with new values otherwise left without change. This is made to avoid wrong coordinates calculation if compensation failed.

[0078] fsr_calc_force_level—this function, indicated by touchscreen control software **135**, is used to set levels (for example minimum level for touch sensitivity) in the touchscreen code in mg when compensation is turned on.

[0079] The only of the above functions used in runtime is fsr_compensate. Other functions are used only during initialization.

[0080] The method described above solves both the linearization, through the conversion of the polynomial curve to a straight line, and the in-between sensor error, through the LM program.

[0081] 2. Temperature Compensation of Sensor Data

[0082] As discussed previously, the performance of the FSR sensors is unfortunately impacted by the surrounding temperature as illustrated in FIG. **5**. Studying the FSR sensor behavior more in detail it is clear that the following conclusions are possible:

- **[0083]** FSR material is NTC (negative temperature coefficient) type resistor material that performs as a negative temperature coefficient (NTC) resistor,
- **[0084]** TCR is dependent on force load level
- **[0085]** Operates both at positive and negative (C) temperature values;
- **[0086]** The Temperature Compensation of Sensor Data can be either in hardware or in software, or a combination.

[0087] One way to describe the curve of an NTC thermistor is to measure the slope of the resistance versus temperature (R/T) curve at one temperature. By definition, the coefficient of resistance is given by:

$$\alpha = \frac{1}{R} * \frac{dR}{dT}$$

[0088] where: T=Temperature in ° C. or K, R=Resistance at Temp T.

[0089] By using an NTC resistor for the pull-down resistor, **22** FIG. **2**, which is matched to the NTC coefficient of the FSR sensor, the temperature dependencies can be eliminated within the existing hardware configuration.

[0090] The alternative method for temperature compensation is to apply software compensation to the data from each sensor as shown in the functional block **16** of FIG. **7**. The temperature program consists of two main components; temperature conditioning (defining the temperature compensation coefficient) depicted in FIG. **9** and the compensation function of FIG. **10**. Defining and recording the temperature coefficient (FIG. **9**) can be done completely outside of the runtime software. The FSR sensor used is measured over the appropriate temperature range (**145**) and the temperature dependent force/voltage data **142** is fed into the coefficient calculation **143**. The coefficient is calculated and stored **141** to be used during runtime as described in FIG. **10**.

[0091] For the compensation function of FIG. **10**, during runtime the FSR sensor device temperature is read during predefined intervals and/or events and as a temperature change is recorded, the event triggers a recalculation of the sensor compensation value **152** based on the pre-defined sensor temperature coefficient.